1 Unveiling Sentiments in Political Speeches: Analyzing the Prime Minister's Address" (PM replies to Motion of No Confidence in Lok Sabha, 10 Aug, 2023)

Importing The Libraries

[40]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

[17]:
```
import warnings
warnings.filterwarnings('ignore')
```

Loading data

[18]:
```
file_path = "/content/pm speech.txt"
```

[19]:
```
with open(file_path , 'r' , encoding = "utf-8") as file:
  speech_text = file.read()
```

NLP comes into the Picture

[20]:
```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

Downloading the necessary batches

[21]:
```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to /root/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[21]:
```
True
```

Performing NLP operations

[22]:
```
speech_text_cleaned = re.sub(r'[^\w\s]' , '' , speech_text)

#This step is often done to ensure that the text is in a consistent case␣
↪(usually lowercase)
# for further processing or analysis.

speech_text_cleaned = speech_text_cleaned.lower()


#Tokenization is the process of splitting a text into individual words or tokens
words = word_tokenize(speech_text_cleaned)

# Stopwords are common words (e.g., "a," "an," "the," "in") that are often␣
↪removed from text
#during natural language processing tasks because they typically don't carry␣
↪significant meaning.

stop_words = set(stopwords.words('english'))

words_filtered = [word for word in words if word not in stop_words]
```

[23]:
```
#For example, it can convert words like "running" to "run" or "better" to "good.
↪"

lemmatizer = WordNetLemmatizer()
words_lemmatized = [lemmatizer.lemmatize(word) for word in words_filtered]
```

Get the average sentiment

[24]:
```
sia = SentimentIntensityAnalyzer()
sentiment_scores = [sia.polarity_scores(word)["compound"] for word in␣
 ↪words_lemmatized]
average_sentiment = sum(sentiment_scores) / len(sentiment_scores)
```

[25]:
```
print("The average sentiment is :" , average_sentiment)
```

```
The average sentiment is : 0.014298377028714108
```

EXTRACT POSITIVE NEGATIVE AND NEUTRAL

[43]:
```
positive_words = [word for i, word in enumerate(words_filtered) if␣
 ↪sentiment_scores[i] > 0.1]
negative_words = [word for i, word in enumerate(words_filtered) if␣
 ↪sentiment_scores[i] < -0.1]
neutral_words = [word for i, word in enumerate(words_filtered) if␣
 ↪sentiment_scores[i] >= -0.1 and sentiment_scores[i] <= 0.1]
```

[44]:
```
print('The positive words are:', positive_words)
```

```
The positive words are: ['gratitude', 'trust', 'free', 'trust', 'fulfill',
'dreams', 'trust', 'confidence', 'top', 'freedom', 'fighters', 'ensure',
'peace', 'assure', 'faith', 'commitment', 'party', 'revered', 'confidence',
'gratitude', 'trust', 'confidence', 'strength', 'lucky', 'confidence',
'blessings', 'better', 'important', 'interest', 'party', 'free', 'energy',
'determination', 'huge', 'dreams', 'strengths', 'dreams', 'free', 'courage',
'opportunity', 'confidence', 'confidence', 'growth', 'trust', 'fulfill',
'dreams', 'marvel', 'helping', 'save', 'helping', 'save', 'helping', 'save',
'trust', 'like', 'wish', 'well', 'best', 'profit', 'increased', 'success',
'growing', 'stronger', 'responsible', 'vision', 'top', 'definite', 'confidence',
'top', 'faith', 'like', 'agree', 'peace', 'trusting', 'trust', 'certain',
'opportunity', 'trust', 'trust', 'confidence', 'help', 'parties', 'faith',
'dwelled', 'fascination', 'freedom', 'fighters', 'dedicated', 'party',
'freebies', 'winning', 'assurances', 'interested', 'great', 'confidence',
'honest', 'ensure', 'assure', 'peace', 'assured', 'assured', 'strong',
'responsible', 'emotional', 'attachment', 'rich', 'goods', 'reached', 'like',
'increased', 'honoured', 'awards', 'hero', 'like', 'celebrated', 'faith',
'commitment', 'assure', 'devote', 'party', 'revered', 'certain', 'devoted',
'trust', 'confidence', 'trust', 'trust', 'inspires', 'credited', 'growing',
'trust', 'growth', 'confidence', 'succeeded', 'strong', 'confidence', 'parties',
'best']
```

[45]:
```
print('The negative words are:', negative_words)
```

```
The negative words are: ['scams', 'poor', 'distrust', 'crimes', 'unacceptable',
'guilty', 'punished', 'pressure', 'stop', 'poor', 'deprived', 'betrayal',
'disappointed', 'scams', 'stressed', 'unsuccessful', 'poor', 'poverty',
'poverty', 'poor', 'poor', 'criticizing', 'distrust', 'bad', 'bad', 'criticism',
'bad', 'misinformation', 'confuse', 'scam', 'crisis', 'severely', 'attacked',
'ills', 'questioned', 'lack', 'poverty', 'hard', 'distrusting', 'lack',
'strike', 'enemy', 'ill', 'misinformed', 'insecurity', 'misinformed', 'low',
'fool', 'arrogance', 'arrogant', 'contradictions', 'damages', 'suffered',
'victims', 'perturbed', 'stuck', 'warned', 'havoc', 'lamented', 'reckless',
'pressure', 'violence', 'saddening', 'crimes', 'unacceptable', 'guilty',
'punished', 'protest', 'failure', 'attack', 'neglect', 'conflict', 'forbidden',
'forbidden', 'loss', 'lack', 'pressure', 'stop', 'worse', 'petty', 'pain',
'suffering']
```

[46]:
```python
print('The neutral words are:', neutral_words)
```

```
The neutral words are: ['come', 'express', 'immense', 'towards', 'every',
'citizen', 'india', 'repeatedly', 'showing', 'government', 'many', 'key',
'legislations', 'get', 'discussion', 'deserved', 'opposition', 'put',
'politics', 'time', 'period', '21st', 'century', 'impact', 'country', 'next',
'thousand', 'years', 'single', 'focus', 'given', 'youth', 'india', 'government',
'today', 'arisen', 'heart', 'opposition', 'able', 'see', 'people', 'steeped',
'2028', 'bring', 'motion', 'country', 'among', '3', 'opposition', 'believes',
'changing', 'names', 'cant', 'change', 'work', 'culture', 'founding', 'fathers',
'country', 'always', 'opposed', 'dynasty', 'politics', 'women', 'central',
'government', 'state', 'government', 'work', 'manipur', 'march', 'path',
'development', 'people', 'manipur', 'mothers', 'daughters', 'manipur', 'nation',
'stands', 'house', 'stands', 'government', 'leave', 'stone', 'unturned',
'manipur', 'gets', 'back', 'track', 'development', 'government', 'given',
'first', 'priority', 'development', 'northeast', 'us', 'sabka', 'saath',
'sabka', 'vishwas', 'slogan', 'article', 'parliament', 'platform', 'parliament',
'highest', 'body', 'country', 'every', 'second', 'utilized', 'country', 'india',
'today', 'crumble', 'india', 'today', 'bend', 'tire', 'prime', 'minister',
'shri', 'narendra', 'modi', 'replied', 'motion', 'lok', 'sabha', 'today',
'addressing', 'house', 'prime', 'minister', 'said', 'come', 'express',
'immense', 'towards', 'every', 'citizen', 'india', 'repeatedly', 'showing',
'government', 'recalled', 'commenting', 'floor', 'test', 'government',
'introduced', 'house', '2018', 'opposition', 'brought', 'noconfidence',
'motion', 'went', 'polls', '2019', 'people', 'declared', 'utmost', 'prime',
'minister', 'said', 'underlined', 'nda', 'bjp', 'seats', 'way', 'prime',
'minister', 'said', 'noconfidence', 'motion', 'introduced', 'opposition',
'government', 'also', 'expressed', 'nda', 'bjp', 'break', 'records', 'come',
'victorious', '2024', 'people', 'prime', 'minister', 'said', 'would',
'opposition', 'participated', 'due', 'seriousness', 'since', 'beginning',
'session', 'mentioned', 'legislations', 'passed', 'past', 'days', 'discussed',
'opposition', 'gave', 'preference', 'politics', 'key', 'legislations', 'many',
'bills', 'linked', 'fishermen', 'data', 'tribals', 'opposition', 'expectations',
'people', 'proven', 'country', 'said', 'prime', 'minister', 'said', 'country',
'watching', 'opposition', 'always', 'people', 'prime', 'minister', 'pointed',
'time', 'comes', 'life', 'nation', 'breaks', 'old', 'shackles', 'moves',
'forward', 'new', 'time', 'period', '21st', 'century', 'time', 'fulfilling',
'aspirations', 'whatever', 'shaped', 'time', 'period', 'impact', 'country',
'next', 'thousand', 'years', 'therefore', 'responsibility', 'single', 'focus',
'development', 'country', 'full', 'dedication', 'realize', 'countrymen',
'emphasized', 'said', 'people', 'youth', 'take', 'us', 'destination',
'continued', '2014', 'later', 'due', 'track', 'record', 'country', 'chose',
'full', 'majority', 'government', 'knew', 'lies', 'capability', 'realizing',
'given', 'youth', 'india', 'government', 'given', 'fly', 'open', 'sky',
'repaired', 'indias', 'standing', 'world', 'taken', 'new', 'heights',
'opposition', 'made', 'attempt', 'break', 'people', 'garb', 'motion', 'said',
'shri', 'modi', 'mentioned', 'startup', 'ecosystem', 'record', 'foreign',
'investment', 'new', 'peaks', 'exports', 'said', 'today', 'arisen', 'heart',
'also', 'talked', 'niti', 'report', '135', 'crore', 'people', 'coming', 'prime',
'minister', 'mentioned', 'imf', 'working', 'paper', 'states', 'india', 'almost',
'eradicated', 'extreme', 'quoting', 'imf', 'prime', 'minister', 'said',
'indian', 'dbt', 'scheme', 'social', 'welfare', 'schemes', 'logistical', 'also',
'quoted', 'states', 'jal', 'jeevan', 'mission', '4', 'lakh', 'lives', 'country',
'swacch', 'bharat', 'abhiyan', '3', 'lakh', 'lives', 'people', 'country',
'reside', 'urban', 'slums', 'added', 'quoting', 'unicef', 'swachh', 'bharat',
'abhiyan', 'prime', 'minister', 'said', 'families', 'country', 'rs', '50000',
'per', 'year', 'ostrich', 'approach', 'opposition', 'prime', 'minister', 'said',
'able', 'see', 'people', 'steeped', 'prime', 'minister', 'said', 'oppositions',
'language', 'constant', 'nitpicking', 'works', 'kala', 'tika', 'ward', 'omen',
'prime', 'minister', 'said', 'target', 'institutions', 'oppositions',
'invariably', 'shine', 'called', 'oppositions', 'secret', 'boon', 'whoever',
'ends', 'said', 'prime', 'minister', 'recalled', 'attitude', 'opposition',
'towards', 'developments', 'banking', 'sector', 'said', 'tried', 'spread',
'people', 'however', 'prime', 'minister', 'interjected', 'net', 'public',
'sector', 'banks', 'twofold', 'also', 'touched', 'upon', 'phone', 'banking',
'pushed', 'country', 'towards', 'npa', 'said', 'country', 'revived', 'moving',
'forward', 'shri', 'modi', 'also', 'gave', 'example', 'hal', 'opposition',
'said', 'hal', 'touching', 'new', 'heights', 'registered', 'highestever',
'revenue', 'throwing', 'light', 'spoken', 'opposition', 'lic', 'prime',
'minister', 'said', 'lic', 'passing', 'day', 'opposition', 'believe',
'capabilities', 'dedication', 'nation', 'prime', 'minister', 'remarked',
'recalled', 'saying', 'days', 'ago', 'third', 'term', 'india', 'become',
'third', 'largest', 'economy', 'world', 'opposition', 'prime', 'minister',
'said', 'government', 'roadmap', 'achieve', 'goal', 'least', 'provided',
'suggestions', 'case', 'called', 'laxity', 'opposition', 'claims', 'nothing',
'needed', 'done', 'become', 'thirdlargest', 'economy', 'world', 'prime',
'minister', 'said', 'approach', 'opposition', 'indicates', 'policies',
'intentions', 'knowhow', 'world', 'economics', 'understanding', 'capabilities',
'india', 'prime', 'minister', 'underlined', 'india', 'sank', 'verge',
'bankruptcy', '1991', 'however', '2014', 'india', 'found', 'place', '5',
'economies', 'world', 'said', 'achieved', 'mantra', 'reform', 'perform',
'transform', 'planning', 'work', 'continue', 'necessary', 'reforms', 'done',
'added', '2028', 'bring', 'motion', 'country', 'among', '3', 'told', 'house',
'continuing', 'approach', 'opposition', 'prime', 'minister', 'talked',
'campaigns', 'swachh', 'bharat', 'jan', 'dhan', 'account', 'yoga', 'ayurveda',
'startup', 'india', 'digital', 'india', 'make', 'india', 'prime', 'minister',
'highlighted', 'infiltration', 'militants', 'kashmir', 'congress', 'rule',
'government', 'would', 'pakistan', 'continue', 'talks', 'simultaneously',
'also', 'touched', 'upon', 'association', 'hurriyat', 'instead', 'kashmiri',
'populace', 'speaking', 'surgical', 'prime', 'minister', 'mentioned',
'opposition', 'chose', 'believe', 'narrative', 'spun', 'instead', 'government',
'issue', 'opposition', 'quick', 'speak', 'country', 'prime', 'minister', 'said',
'mentioned', 'report', 'foreign', 'agency', 'touted', 'nation', 'dealing',
'food', 'ahead', 'india', 'parameters', 'said', 'opposition', 'latches',
'reports', 'tries', 'defame', 'country', 'every', 'gets', 'also', 'gave',
'example', 'madeinindia', 'corona', 'vaccine', 'said', 'opposition', 'instead',
'looked', 'towards', 'foreignmade', 'vaccines', 'underlined', 'opposition',
'capabilities', 'india', 'people', 'similarly', 'level', 'opposition', 'eyes',
'people', 'extreme', 'prime', 'minister', 'also', 'said', 'cosmetic', 'changes',
'alliance', 'building', 'people', 'country', 'simple', 'change', 'name',
'change', 'fortune', 'opposition', 'alliance', 'taken', 'nda', 'survive',
'added', 'two', 'first', 'ego', '26', 'second', 'ego', 'one', 'family', 'even',
'splintered', 'india', 'india', 'said', 'opposition', 'believes', 'changing',
'names', 'cant', 'change', 'work', 'culture', 'emphasized', 'referring',
'divisive', 'comment', 'minister', 'tamil', 'nadu', 'government', 'prime',
'minister', 'reiterated', 'state', 'said', 'tamil', 'nadu', 'state', 'stream',
'patriotism', 'flows', 'continuously', 'prime', 'minister', 'opposition',
'names', 'mentioned', 'every', 'scheme', 'key', 'marker', 'named', 'members',
'one', 'family', 'prime', 'minister', 'called', 'india', 'ghamndia',
'coalition', 'coalition', 'underlined', 'among', 'partners', 'shri', 'modi',
'emphasized', 'founding', 'fathers', 'country', 'always', 'opposed', 'dynasty',
'politics', 'dynasty', 'system', 'common', 'citizen', 'key', 'leaders', 'due',
'dynasty', 'politics', 'said', 'said', 'many', 'portraits', 'stalwarts', 'type',
'politics', 'found', 'place', 'parliament', 'later', 'years', 'noncongress',
'governments', 'also', 'mentioned', 'statue', 'unity', 'pradhanmantri',
'sangrahalaya', 'museum', 'prime', 'ministers', 'rises', 'politics', 'prime',
'minister', 'reiterated', 'even', 'though', 'people', 'india', 'elected',
'full', 'majority', 'government', 'twice', '30', 'years', 'opposition', 'garib',
'ka', 'beta', 'sitting', 'prime', 'ministers', 'chair', 'pointed', 'misuse',
'aircrafts', 'naval', 'vessels', 'past', 'opposition', 'rectified',
'transportation', 'vaccines', 'bringing', 'back', 'foreign', 'lands', 'prime',
'minister', 'politics', 'cited', 'situation', 'neighboring', 'countries',
'example', 'politics', 'bring', 'tendency', 'elections', 'people', 'put',
'tremendous', 'development', 'projects', 'shelved', 'prime', 'minister', 'said',
'opposition', 'never', 'discussing', 'manipur', 'situation', 'said', 'home',
'minister', 'explained', 'issues', 'detail', 'patience', 'without', 'politics',
'explanation', 'home', 'minister', 'effort', 'convey', 'concern', 'country',
'nation', 'attempt', 'convey', 'houses', 'manipur', 'effort', 'discuss', 'find',
'ways', 'speaking', 'manipur', 'issue', 'prime', 'minister', 'said', 'manipur',
'women', 'central', 'government', 'state', 'government', 'work', 'people',
'india', 'basis', 'effort', 'making', 'manipur', 'coming', 'times', 'prime',
'minister', 'said', 'people', 'manipur', 'mothers', 'daughters', 'manipur',
'nation', 'stands', 'house', 'stands', 'also', 'government', 'leave', 'stone',
'unturned', 'manipur', 'gets', 'back', 'track', 'development', 'prime',
'minister', 'registered', 'use', 'objectionable', 'language', 'maa', 'bharati',
'house', 'said', 'people', 'partition', 'even', 'berated', 'vande', 'mataram',
'shri', 'modi', 'also', 'mentioned', 'kachchatheevu', 'issue', 'example',
'opposition', 'prime', 'minister', 'mentioned', 'three', 'incidents',
'regarding', 'northeast', 'first', '5th', 'march', '1966', 'airforce', 'used',
'people', 'mizoram', 'second', 'radio', 'transmission', 'prime', 'minister',
'nehru', '1962', 'people', 'northeast', 'left', 'fend', 'chinese', 'invasion',
'also', 'cited', 'ram', 'manohar', 'lohias', 'allegation', 'region', 'prime',
'minister', 'informed', 'current', 'government', 'ministers', 'done', '400',
'night', 'stays', 'various', 'district', 'headquarters', 'northeast', 'prime',
'minister', 'visited', '50', 'times', 'northeast', 'even', 'becoming', 'pm',
'traveled', 'across', 'region', 'shri', 'modi', 'said', 'prime', 'minister',
'reiterated', 'situation', 'manipur', 'presented', 'way', 'arose', 'recently',
'root', 'cause', 'issues', 'manipur', 'congress', 'politics', 'manipur',
'filled', 'indian', 'culture', 'heritage', 'manipur', 'land', 'innumerable',
'sacrifices', 'said', 'recalled', 'time', 'congress', 'government', 'state',
'every', 'institution', 'operated', 'beck', 'call', 'extremist',
'organizations', 'putting', 'photograph', 'mahatma', 'gandhi', 'government',
'offices', 'also', 'mentioned', 'bombing', 'statue', 'netaji', 'subhas',
'chandra', 'bose', 'museum', 'azad', 'hind', 'fauj', 'moirang', 'recalled',
'singing', 'national', 'anthem', 'schools', 'manipur', 'campaign', 'initiated',
'burn', 'books', 'libraries', 'prime', 'minister', 'gave', 'several',
'examples', 'extremist', 'activities', 'region', 'congress', 'rule',
'mentioned', 'temples', 'shutting', 'doors', '4', 'evening', 'bombing',
'iskcon', 'temple', 'imphal', 'led', 'lives', 'protection', 'money', 'paid',
'extremists', 'government', 'officers', 'prime', 'minister', 'said', 'coming',
'days', 'northeast', 'going', 'center', 'development', 'said', 'aware', 'fact',
'movements', 'global', 'system', 'bring', 'change', 'southeast', 'asia',
'asean', 'countries', 'impact', 'northeast', 'prime', 'minister', 'said',
'government', 'given', 'first', 'priority', 'development', 'northeast', 'shri',
'modi', 'talked', 'investment', 'infrastructure', 'northeast', 'mentioned',
'modern', 'highways', 'railways', 'airports', 'becoming', 'identity',
'northeast', 'agartala', 'got', 'connected', 'rail', 'connectivity', 'first',
'time', 'train', 'manipur', 'first', 'time', 'first', 'time', 'modern', 'train',
'vande', 'bharat', 'ran', 'region', 'first', 'greenfield', 'airport',
'constructed', 'arunachal', 'pradesh', 'sikkim', 'got', 'connected', 'air',
'travel', 'first', 'time', 'aiims', 'opened', 'northeast', 'national', 'sports',
'university', 'opened', 'manipur', 'indian', 'institute', 'mass',
'communication', 'mizoram', 'first', 'time', 'northeasts', 'participation',
'council', 'ministers', 'first', 'time', 'woman', 'represented', 'nagaland',
'rajya', 'sabha', 'first', 'time', 'many', 'people', 'northeast', 'padma',
'lachit', 'burfukan', 'republic', 'day', 'museum', 'name', 'rani', 'gaidinliu',
'established', 'said', 'us', 'sabka', 'saath', 'sabka', 'vishwas', 'slogan',
'article', 'said', 'prime', 'minister', 'adding', 'people', 'country', 'every',
'particle', 'body', 'every', 'moment', 'service', 'countrymen', 'prime',
'minister', 'emphasized', 'parliament', 'platform', 'parliament', 'highest',
'body', 'country', 'therefore', 'imperative', 'parliamentarians', 'seriousness',
'much', 'resources', 'every', 'second', 'utilized', 'country', 'added',
'seriousness', 'one', 'politics', 'country', 'run', 'last', '9', 'years',
'prime', 'minister', 'said', 'common', 'citizens', 'soaring', 'new', 'heights',
'every', 'indian', 'filled', 'india', 'today', 'crumble', 'india', 'today',
'bend', 'tire', 'shri', 'modi', 'said', 'urged', 'citizens', 'move', 'forward',
'resolution', 'said', 'common', 'people', 'world', 'believe', 'india', 'world',
'india', 'common', 'citizens', 'past', 'years', 'prime', 'minister', 'said',
'government', 'laying', 'foundations', 'viksit', 'bharat', 'expressed',
'foundation', 'lead', 'india', 'become', 'developed', 'nation', 'year', '2047',
'underlined', 'nation', 'come', 'situations', 'together', 'urged', 'political',
'misuse', 'land', 'manipur', 'politics', 'must', 'empathize', 'recovery', 'way',
'forward', 'prime', 'minister', 'appealed']
```

Freqdist = It helps you count the occurrences of each unique item in the list and provides various methods for analyzing and visualizing these frequencies.

[47]:
```python
word_freq_positive = nltk.FreqDist(positive_words)
word_freq_negative = nltk.FreqDist(negative_words)
word_freq_neutral = nltk.FreqDist(neutral_words)
```

[48]:
```python
print('The positive words frequency is:', word_freq_positive)
```

```
The positive words frequency is: <FreqDist with 74 samples and 138 outcomes>
```

[49]:
```python
print('The negative words frequency is:', word_freq_negative)
```

```
The negative words frequency is: <FreqDist with 61 samples and 82 outcomes>
```

[50]:
```python
print('The neutral words frequency is:', word_freq_neutral)
```

```
The neutral words frequency is: <FreqDist with 658 samples and 1382 outcomes>
```

DATA VISUALIZATION

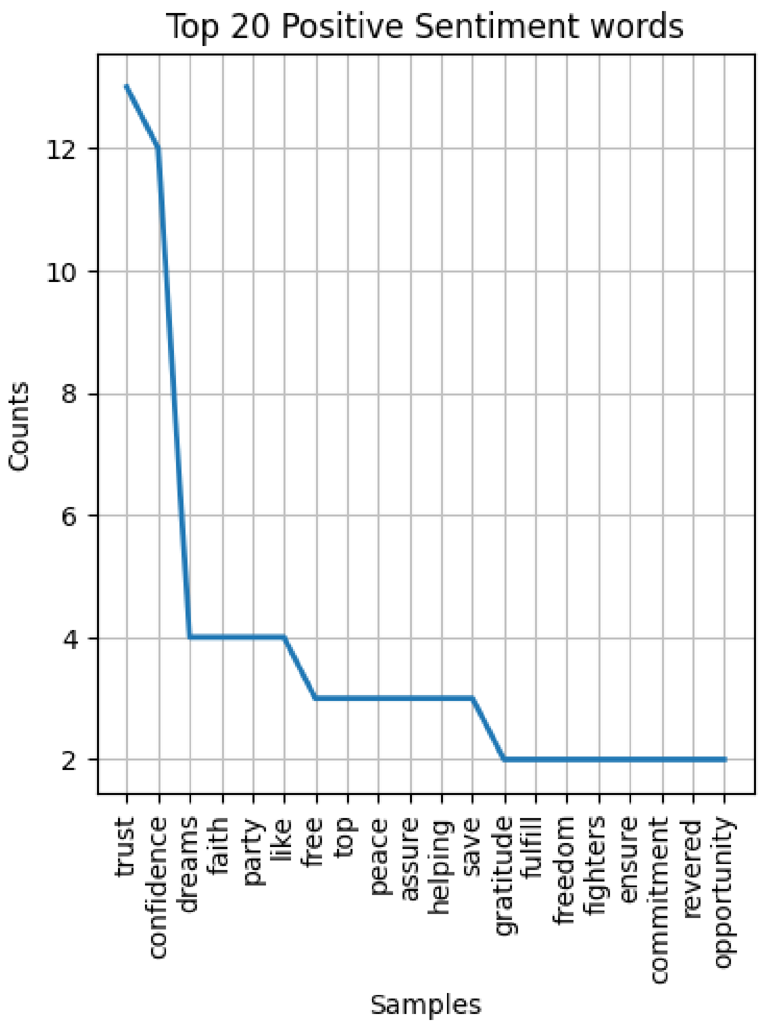
PLOTTING THESE OCCURRENCES for Visualization

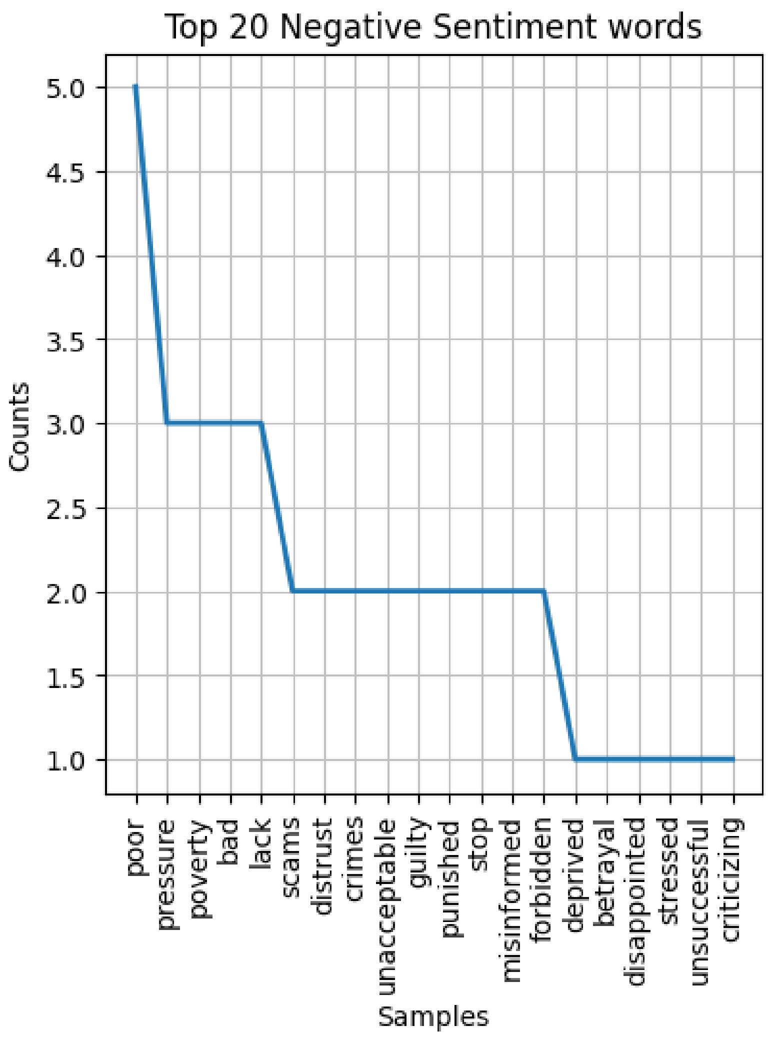
subplot - means plot within plot

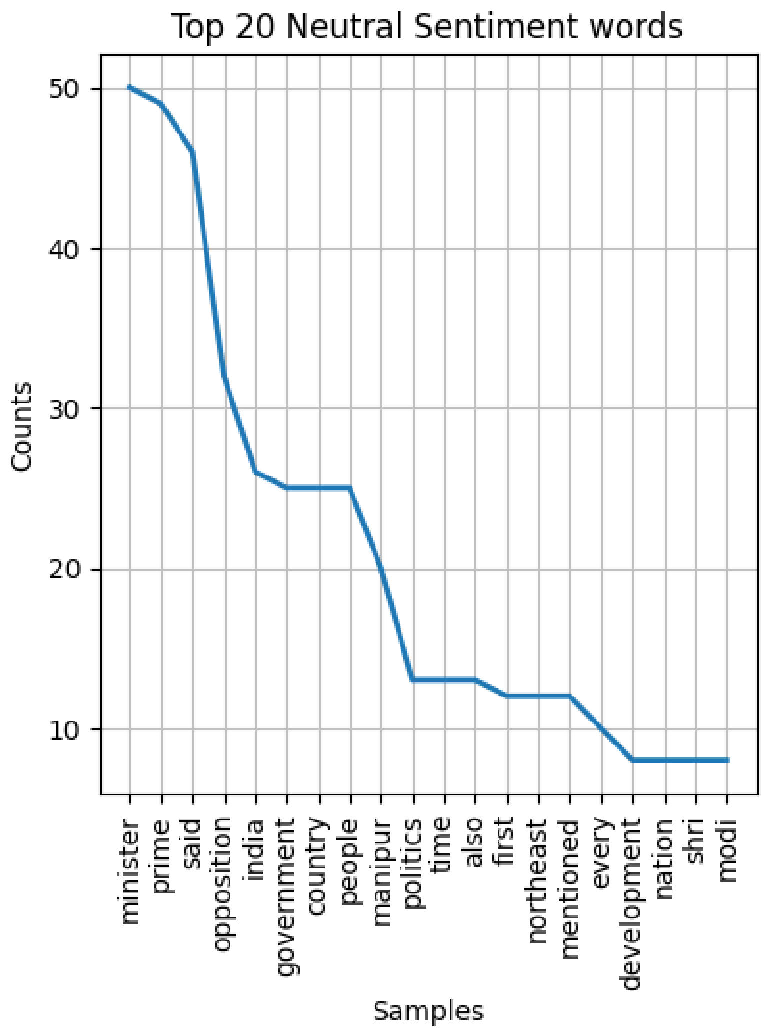
eg : plt.subplot(133) creates a subplot in a figure with a grid layout of 1 row and 3 columns, and it positions the subplot in the third (rightmost) column.(same for all)

[51]:
```python
plt.figure(figsize=(15,5))
plt.subplot(131)
word_freq_positive.plot(20,title="Top 20 Positive Sentiment words")

plt.figure(figsize=(15,5))
plt.subplot(132)
word_freq_negative.plot(20,title="Top 20 Negative Sentiment words")

plt.figure(figsize=(15,5))
plt.subplot(133)
word_freq_neutral.plot(20,title="Top 20 Neutral Sentiment words")
```

```
plt.tight_layout()
plt.show()
```




Top 20 Negative Sentiment words
Counts
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
poor
pressure
poverty
bad
lack
scams
distrust
crimes
unacceptable
guilty
punished
stop
misinformed
forbidden
deprived
betrayal
disappointed
stressed
unsuccessful
criticizing
Samples



```
<Figure size 640x480 with 0 Axes>
```

Generating NeW Data Frames

```
[52]: df_positive = pd.DataFrame(word_freq_positive.most_common(20), columns=['Word',␣
        ↪'Frequency'])
      df_negative = pd.DataFrame(word_freq_negative.most_common(20), columns=['Word',␣
        ↪'Frequency'])
      df_neutral = pd.DataFrame(word_freq_neutral.most_common(20), columns=['Word',␣
        ↪'Frequency'])
```

Plotting BAR GRaphs

```
[53]: fig_positive = px.bar(df_positive, x='Word', y='Frequency', title="Top 20␣
        ↪Positive Sentiment Words")
```

```
fig_negative = px.bar(df_negative, x='Word', y='Frequency', title="Top 20␣
  ↪Negative Sentiment Words")
fig_neutral = px.bar(df_neutral, x='Word', y='Frequency', title="Top 20 Neutral␣
  ↪Sentiment Words")

fig_positive.show()
fig_negative.show()
fig_neutral.show()
```

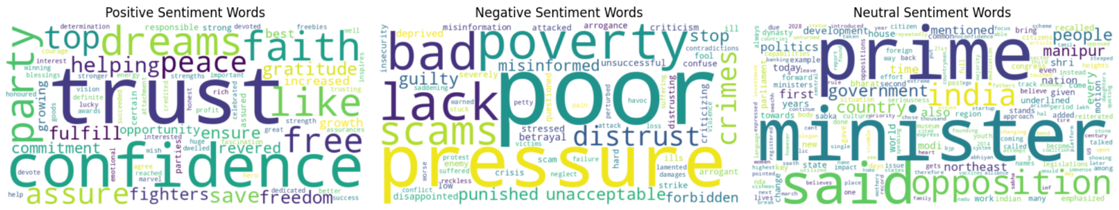
PLOTTING WORD CLOUDS

[56]:
```
wordcloud_positive = WordCloud(width=800, height=400, background_color="white").
  ↪generate_from_frequencies(word_freq_positive)
wordcloud_negative = WordCloud(width=800, height=400, background_color="white").
  ↪generate_from_frequencies(word_freq_negative)
wordcloud_neutral = WordCloud(width=800, height=400, background_color="white").
  ↪generate_from_frequencies(word_freq_neutral)
```

Bilinear Interpolation:

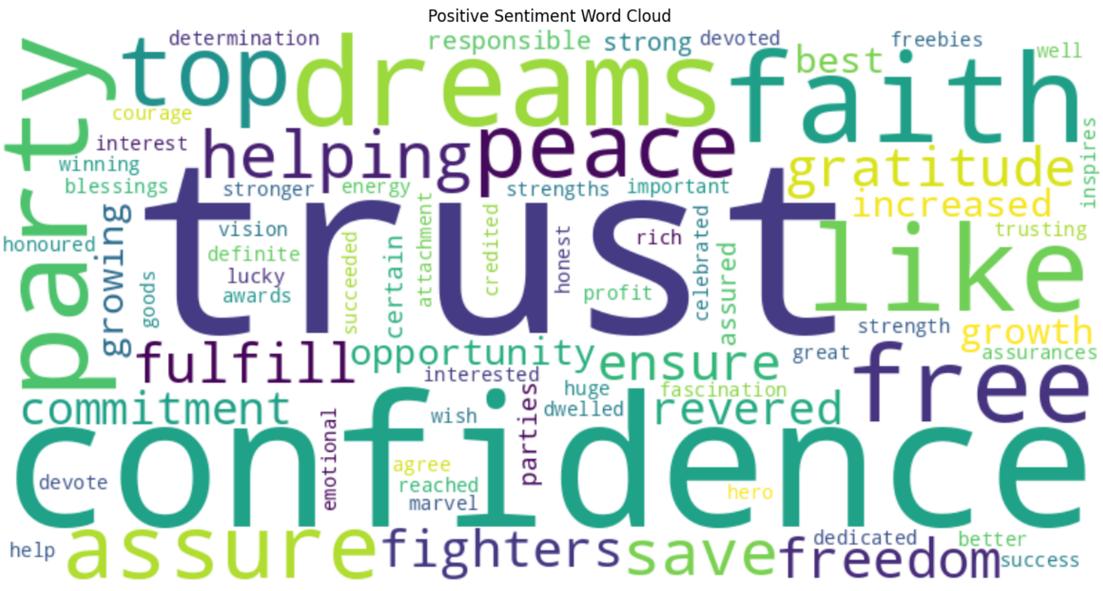
It takes the weighted average of the four nearest known pixels to estimate the value of the unknown pixel. This method creates smoother transitions and is commonly used in image resizing.

[57]:
```
plt.figure(figsize=(15, 5))

plt.subplot(131)
plt.imshow(wordcloud_positive, interpolation="bilinear")
plt.axis("off")
plt.title("Positive Sentiment Words")

plt.subplot(132)
plt.imshow(wordcloud_negative, interpolation="bilinear")
plt.axis("off")
plt.title("Negative Sentiment Words")

plt.subplot(133)
plt.imshow(wordcloud_neutral, interpolation="bilinear")
plt.axis("off")
plt.title("Neutral Sentiment Words")

plt.tight_layout()
plt.show()
```



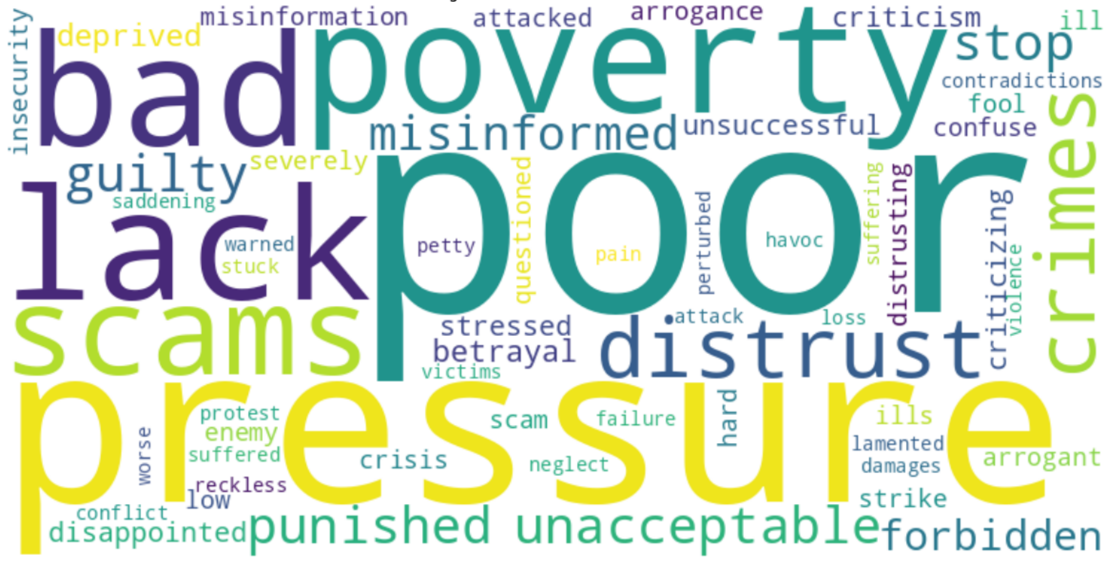
Seperately

```
[58]: plt.figure(figsize=(15, 10))
      plt.imshow(wordcloud_positive, interpolation="bilinear")
      plt.title("Positive Sentiment Word Cloud")
      plt.axis("off")
      plt.show()
```



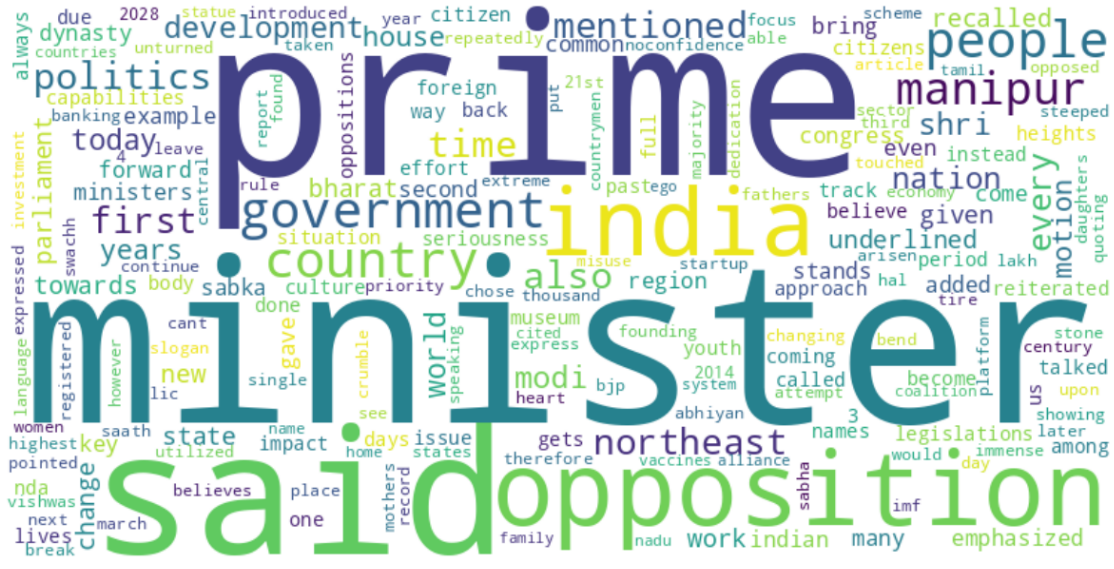
```
[60]: plt.figure(figsize=(15, 10))
      plt.imshow(wordcloud_negative, interpolation="bilinear")
      plt.title("Negative Sentiment Word Cloud")
      plt.axis("off")
      plt.show()
```

Negative Sentiment Word Cloud



```
[61]: plt.figure(figsize=(15, 10))
      plt.imshow(wordcloud_neutral, interpolation="bilinear")
      plt.title("Neutral Sentiment Word Cloud")
      plt.axis("off")
      plt.show()
```

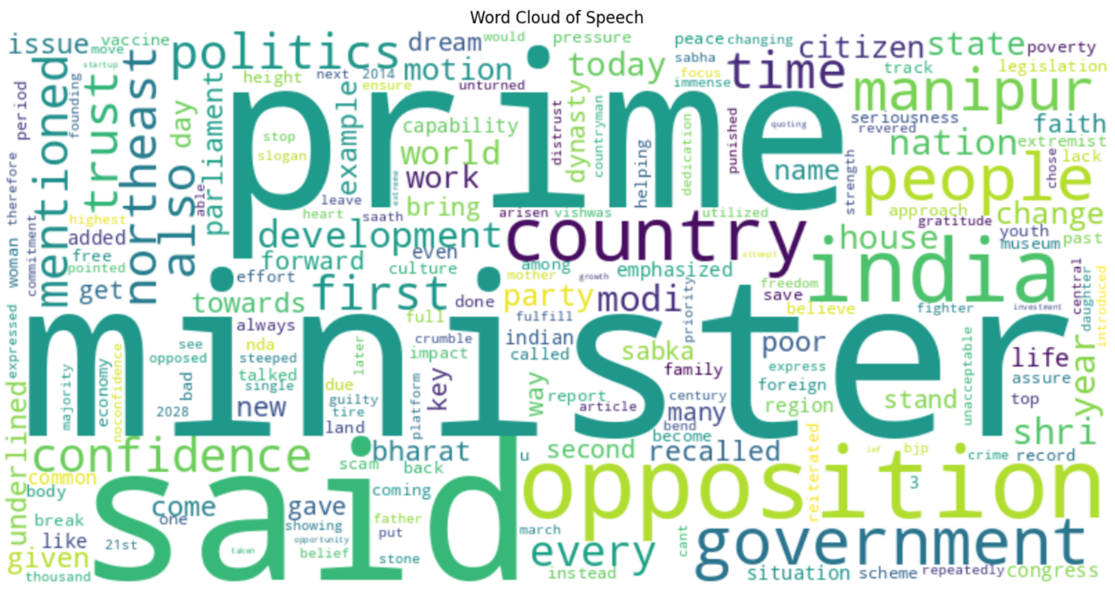
Neutral Sentiment Word Cloud



```
[62]: word_freq = nltk.FreqDist(words_lemmatized)
```

```
wordcloud = WordCloud(width = 800 , height = 400 , background_color="white").
 ↪generate_from_frequencies(word_freq)
```

```
[64]: plt.figure(figsize=(15, 10))
      plt.imshow(wordcloud, interpolation="bilinear")
      plt.axis("off")
      plt.title("Word Cloud of Speech")
      plt.show()
```



Summazrizing Percentages

```
[65]: total_words = len(words_filtered)
      positive_percentage = (len(positive_words) / total_words) * 100
      negative_percentage = (len(negative_words) / total_words) * 100
      neutral_percentage = (len(neutral_words) / total_words) * 100

      print("Positive Sentiment Percentage:", positive_percentage)
      print("Negative Sentiment Percentage:", negative_percentage)
      print("Neutral Sentiment Percentage:", neutral_percentage)
```

```
Positive Sentiment Percentage: 8.614232209737828
Negative Sentiment Percentage: 5.118601747815231
Neutral Sentiment Percentage: 86.26716604244695
```

Creating new Data

```
[66]: data = {"Sentiment" : ['Positive' , 'Negative' , 'Neutral'] ,
              "Percentage" : [positive_percentage , negative_percentage ,␣
       ↪neutral_percentage]
```

```
        }
```

Creating new Data Frame

```
[68]: df_percentages = pd.DataFrame(data)
```

```
[70]: df_percentages
```

```
[70]:   Sentiment  Percentage
      0  Positive    8.614232
      1  Negative    5.118602
      2   Neutral   86.267166
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c410f54dc00>
```

```
import numpy as np
from google.colab import autoviz

def value_plot(df, y, figscale=1):
  from matplotlib import pyplot as plt
  df[y].plot(kind='line', figsize=(8 * figscale, 4 * figscale), title=y)
  plt.gca().spines[['top', 'right']].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = value_plot(df_percentages, *['Percentage'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c410fa00b50>
```

```
import numpy as np
from google.colab import autoviz

def histogram(df, colname, num_bins=20, figscale=1):
  from matplotlib import pyplot as plt
  df[colname].plot(kind='hist', bins=num_bins, title=colname,␣
 ↪figsize=(8*figscale, 4*figscale))
  plt.gca().spines[['top', 'right',]].set_visible(False)
  plt.tight_layout()
  return autoviz.MplChart.from_current_mpl_state()

chart = histogram(df_percentages, *['Percentage'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c410f8b9300>
```

```
import numpy as np
from google.colab import autoviz

def categorical_histogram(df, colname, figscale=1, mpl_palette_name='Dark2'):
  from matplotlib import pyplot as plt
  import seaborn as sns
  df.groupby(colname).size().plot(kind='barh', color=sns.palettes.
 ↪mpl_palette(mpl_palette_name), figsize=(8*figscale, 4.8*figscale))
  plt.gca().spines[['top', 'right',]].set_visible(False)
  return autoviz.MplChart.from_current_mpl_state()

chart = categorical_histogram(df_percentages, *['Sentiment'], **{})
chart
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7c410f581d20>
```

```
import numpy as np
from google.colab import autoviz

def violin_plot(df, value_colname, facet_colname, figscale=1,␣
 ↪mpl_palette_name='Dark2', **kwargs):
  from matplotlib import pyplot as plt
  import seaborn as sns
  figsize = (12 * figscale, 1.2 * figscale * len(df[facet_colname].unique()))
  plt.figure(figsize=figsize)
  sns.violinplot(df, x=value_colname, y=facet_colname, palette=mpl_palette_name,␣
 ↪**kwargs)
  sns.despine(top=True, right=True, bottom=True, left=True)
  return autoviz.MplChart.from_current_mpl_state()

chart = violin_plot(df_percentages, *['Percentage', 'Sentiment'], **{'inner':␣
 ↪'stick'})
chart
```

Plotting Overall Graph

```
[71]: fig = px.bar(df_percentages, x='Sentiment', y='Percentage', color='Sentiment',
                   labels={'Sentiment': 'Sentiment Category', 'Percentage':␣
       ↪'Percentage (%)'},
                   title='Percentage of Words in Each Sentiment Category')
      fig.show()
```

What is Gensim?:

Gensim is an open-source Python library designed for natural language processing (NLP) and machine learning. It is specifically focused on unsupervised topic modeling, document similarity analysis, and other tasks related to text and document analysis

The gensim.corpora module within Gensim is responsible for managing text corpora. A corpus is a collection of text documents, and Gensim provides tools to represent and work with corpora in a way that is suitable for various NLP tasks, particularly topic modeling.

Latent Dirichlet Allocation (LDA) models:

LdaModel class allows you to create, train, and use LDA models on text corpora. LDA models are used to discover hidden topics in a collection of documents and assign topic probabilities to individual documents and words.

[72]:
```
import gensim
from gensim import corpora
from gensim.models.ldamodel import LdaModel
```

he corpora.Dictionary function creates a dictionary that assigns a unique ID

to each unique word in the provided list(s) of words.

[73]:
```
dictionary = corpora.Dictionary([words_filtered])
```

The doc2bow() method of the dictionary object is used to convert a document (represented as a list of words) into a bag of words. “BOW” stands for “bag of words,” which is a common method for representing text data in a format suitable for various natural language processing tasks.

[74]:
```
corpus = [dictionary.doc2bow(words_filtered)]
```

Gensim’s Latent Dirichlet Allocation (LDA) model to perform topic modeling on a corpus of documents.

[76]:
```
lda_model = LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15)

topics = lda_model.print_topics(num_words=5)
for topic in topics:
    print(topic)
```

```
(0, '0.029*"minister" + 0.028*"prime" + 0.026*"said" + 0.018*"opposition" +
0.015*"india"')
(1, '0.001*"prime" + 0.001*"minister" + 0.001*"country" + 0.001*"said" +
0.001*"india"')
(2, '0.001*"said" + 0.001*"minister" + 0.001*"prime" + 0.001*"opposition" +
0.001*"government"')
(3, '0.001*"prime" + 0.001*"minister" + 0.001*"said" + 0.001*"opposition" +
0.001*"india"')
(4, '0.001*"minister" + 0.001*"prime" + 0.001*"said" + 0.001*"opposition" +
0.001*"india"')
```

For example, if your speech_text contains sentences like “Apple Inc. is headquartered in Cupertino, California,” this code will identify “Apple Inc.” as an entity with the label “ORG” (organization) and “Cupertino, California” as an entity with the label “GPE” (geopolitical entity). The output will display these entities and their labels.

[77]:
```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp(speech_text)
entities = [(ent.text, ent.label_) for ent in doc.ents]
```

```python
for entity, label in entities:
    print(f"Entity: {entity}, Label: {label}")
```

```
Entity: India, Label: GPE
Entity: the 21st century, Label: DATE
Entity: the next thousand years, Label: DATE
Entity: India, Label: GPE
Entity: Today, Label: DATE
Entity: 2028, Label: DATE
Entity: Country, Label: ORG
Entity: 3, Label: CARDINAL
Entity: the Central Government, Label: ORG
Entity: the State Government, Label: ORG
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: House, Label: ORG
Entity: Manipur, Label: GPE
Entity: first, Label: ORDINAL
Entity: Northeast, Label: LOC
Entity: Sabka Saath Sabka Vishwas, Label: PERSON
Entity: Parliament, Label: ORG
Entity: Party, Label: ORG
Entity: Parliament, Label: ORG
Entity: The India of today, Label: WORK_OF_ART
Entity: India, Label: GPE
Entity: Shri Narendra Modi, Label: PERSON
Entity: the Motion of No Confidence, Label: ORG
Entity: Lok Sabha, Label: PERSON
Entity: today, Label: DATE
Entity: House, Label: ORG
Entity: India, Label: GPE
Entity: 2018, Label: DATE
Entity: 2019, Label: DATE
Entity: NDA, Label: ORG
Entity: BJP, Label: ORG
Entity: NDA, Label: ORG
Entity: BJP, Label: ORG
Entity: 2024, Label: DATE
Entity: the past few days, Label: DATE
Entity: the 21st century, Label: DATE
Entity: the next thousand years, Label: DATE
Entity: 2014, Label: DATE
Entity: India, Label: GPE
Entity: India, Label: GPE
Entity: Shri Modi, Label: ORG
Entity: Today, Label: DATE
```

```
Entity: NITI, Label: GPE
Entity: about 13.5, Label: CARDINAL
Entity: India, Label: GPE
Entity: Indian, Label: NORP
Entity: WHO, Label: ORG
Entity: the Jal Jeevan Mission, Label: ORG
Entity: 4, Label: CARDINAL
Entity: the Swacch Bharat Abhiyan, Label: NORP
Entity: 3, Label: CARDINAL
Entity: the Swachh Bharat Abhiyan, Label: ORG
Entity: Rs 50,000, Label: PRODUCT
Entity: a ‘Kala Tika, Label: FAC
Entity: Shri Modi, Label: LAW
Entity: HAL, Label: ORG
Entity: HAL, Label: PERSON
Entity: LIC, Label: ORG
Entity: LIC, Label: ORG
Entity: a few days ago, Label: DATE
Entity: third, Label: ORDINAL
Entity: India, Label: GPE
Entity: third, Label: ORDINAL
Entity: third, Label: ORDINAL
Entity: India, Label: GPE
Entity: India, Label: GPE
Entity: 1991, Label: DATE
Entity: 2014, Label: DATE
Entity: India, Label: GPE
Entity: 5, Label: CARDINAL
Entity: 2028, Label: DATE
Entity: Country, Label: ORG
Entity: 3, Label: CARDINAL
Entity: House, Label: ORG
Entity: Swachh, Label: NORP
Entity: Jan Dhan Account, Label: PERSON
Entity: Yoga, Ayurveda, Label: ORG
Entity: India, Label: GPE
Entity: Digital India, Label: ORG
Entity: Make, Label: GPE
Entity: India, Label: GPE
Entity: Kashmir, Label: LOC
Entity: Congress, Label: ORG
Entity: Pakistan, Label: GPE
Entity: Kashmiri, Label: ORG
Entity: India, Label: GPE
Entity: India, Label: GPE
Entity: India, Label: GPE
Entity: NDA, Label: ORG
Entity: two, Label: CARDINAL
```

```
Entity: first, Label: ORDINAL
Entity: 26, Label: CARDINAL
Entity: second, Label: ORDINAL
Entity: India, Label: GPE
Entity: I.N.D.I.A., Label: GPE
Entity: the Tamil Nadu Government, Label: ORG
Entity: Tamil Nadu, Label: PERSON
Entity: one, Label: CARDINAL
Entity: a ‘Ghamndia, Label: ORG
Entity: Shri Modi, Label: LAW
Entity: Parliament, Label: ORG
Entity: the later years, Label: DATE
Entity: non-Congress, Label: ORG
Entity: the Statue of Unity, Label: FAC
Entity: Museum, Label: ORG
Entity: India, Label: GPE
Entity: 30 years, Label: DATE
Entity: Manipur, Label: GPE
Entity: Home, Label: ORG
Entity: Home, Label: ORG
Entity: House, Label: ORG
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: the Central Government, Label: ORG
Entity: the State Government, Label: ORG
Entity: India, Label: GPE
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: House, Label: ORG
Entity: Government, Label: ORG
Entity: Manipur, Label: GPE
Entity: Maa Bharati, Label: PERSON
Entity: House, Label: ORG
Entity: Partition, Label: ORG
Entity: Vande Mataram, Label: ORG
Entity: Shri Modi, Label: ORG
Entity: Kachchatheevu, Label: FAC
Entity: three, Label: CARDINAL
Entity: Northeast, Label: LOC
Entity: First, Label: ORDINAL
Entity: 5th March 1966, Label: DATE
Entity: Airforce, Label: PRODUCT
Entity: Mizoram, Label: GPE
Entity: Second, Label: ORDINAL
Entity: Nehru, Label: PERSON
Entity: 1962, Label: DATE
```

```
Entity: Northeast, Label: LOC
Entity: Chinese, Label: NORP
Entity: 400 night, Label: TIME
Entity: Northeast, Label: LOC
Entity: 50, Label: CARDINAL
Entity: Northeast, Label: LOC
Entity: Shri Modi, Label: LAW
Entity: Manipur, Label: GPE
Entity: Manipur, Label: GPE
Entity: Congress, Label: ORG
Entity: Manipur, Label: GPE
Entity: Indian, Label: NORP
Entity: Manipur, Label: GPE
Entity: Congress, Label: ORG
Entity: Mahatma Gandhi, Label: PERSON
Entity: Netaji Subhas, Label: PERSON
Entity: Chandra Bose, Label: PERSON
Entity: the Museum of Azad Hind Fauj, Label: ORG
Entity: Moirang, Label: GPE
Entity: the National Anthem, Label: ORG
Entity: Manipur, Label: GPE
Entity: Congress, Label: ORG
Entity: 4 in the evening, Label: TIME
Entity: Iskcon, Label: NORP
Entity: Imphal, Label: GPE
Entity: the coming days, Label: DATE
Entity: Northeast, Label: LOC
Entity: South-East Asia, Label: LOC
Entity: ASEAN, Label: ORG
Entity: Northeast, Label: LOC
Entity: first, Label: ORDINAL
Entity: Northeast, Label: LOC
Entity: Shri Modi, Label: LAW
Entity: Northeast, Label: LOC
Entity: Northeast, Label: LOC
Entity: Agartala, Label: PERSON
Entity: first, Label: ORDINAL
Entity: Manipur, Label: GPE
Entity: first, Label: ORDINAL
Entity: first, Label: ORDINAL
Entity: Vande Bharat, Label: GPE
Entity: first, Label: ORDINAL
Entity: Arunachal Pradesh, Label: ORG
Entity: Sikkim, Label: PERSON
Entity: first, Label: ORDINAL
Entity: AIIMS, Label: ORG
Entity: Northeast, Label: LOC
Entity: National Sports University, Label: ORG
```

```
Entity: Manipur, Label: GPE
Entity: Indian Institute of Mass Communication, Label: ORG
Entity: Mizoram, Label: GPE
Entity: first, Label: ORDINAL
Entity: Northeast, Label: LOC
Entity: the Council of Ministers, Label: ORG
Entity: first, Label: ORDINAL
Entity: Nagaland, Label: GPE
Entity: the Rajya Sabha, Label: FAC
Entity: first, Label: ORDINAL
Entity: Northeast, Label: LOC
Entity: Padma Awards, Label: ORG
Entity: Lachit Burfukan, Label: PERSON
Entity: Republic Day, Label: DATE
Entity: Rani Gaidinliu, Label: PERSON
Entity: Sabka Saath Sabka Vishwas, Label: PERSON
Entity: Parliament, Label: ORG
Entity: Party, Label: ORG
Entity: Parliament, Label: ORG
Entity: Parliamentarians, Label: NORP
Entity: the last 9 years, Label: DATE
Entity: Indian, Label: NORP
Entity: India, Label: GPE
Entity: Shri Modi, Label: LAW
Entity: India, Label: GPE
Entity: India, Label: GPE
Entity: the past few years, Label: DATE
Entity: Viksit Bharat, Label: PERSON
Entity: India, Label: GPE
Entity: the year 2047, Label: DATE
Entity: Manipur, Label: GPE
```

KeyBERT is a library for keyword extraction and allows you to find important words or phrases in a text.

```
[79]: !pip install keybert
```

```
Collecting keybert
  Downloading keybert-0.7.0.tar.gz (21 kB)
  Preparing metadata (setup.py) … done
Collecting sentence-transformers>=0.3.8 (from keybert)
  Downloading sentence-transformers-2.2.2.tar.gz (85 kB)
                           86.0/86.0 kB
2.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: scikit-learn>=0.22.2 in
/usr/local/lib/python3.10/dist-packages (from keybert) (1.2.2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-
```

```
packages (from keybert) (1.23.5)
Requirement already satisfied: rich>=10.4.0 in /usr/local/lib/python3.10/dist-
packages (from keybert) (13.5.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich>=10.4.0->keybert) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich>=10.4.0->keybert) (2.16.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=0.22.2->keybert) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=0.22.2->keybert) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22.2->keybert)
(3.2.0)
Collecting transformers<5.0.0,>=4.6.0 (from sentence-
transformers>=0.3.8->keybert)
  Downloading transformers-4.33.1-py3-none-any.whl (7.6 MB)
                           7.6/7.6 MB
15.6 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from sentence-transformers>=0.3.8->keybert) (4.66.1)
Requirement already satisfied: torch>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from sentence-transformers>=0.3.8->keybert) (2.0.1+cu118)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-
packages (from sentence-transformers>=0.3.8->keybert) (0.15.2+cu118)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(from sentence-transformers>=0.3.8->keybert) (3.8.1)
Collecting sentencepiece (from sentence-transformers>=0.3.8->keybert)
  Downloading
sentencepiece-0.1.99-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(1.3 MB)
                           1.3/1.3 MB
11.8 MB/s eta 0:00:00
Collecting huggingface-hub>=0.4.0 (from sentence-
transformers>=0.3.8->keybert)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
                          268.8/268.8 kB
25.8 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.4.0->sentence-transformers>=0.3.8->keybert)
(3.12.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from huggingface-hub>=0.4.0->sentence-transformers>=0.3.8->keybert) (2023.6.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.4.0->sentence-transformers>=0.3.8->keybert)
(2.31.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.4.0->sentence-transformers>=0.3.8->keybert)
```

```
(6.0.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-
transformers>=0.3.8->keybert) (4.5.0)
Requirement already satisfied: packaging>=20.9 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-
transformers>=0.3.8->keybert) (23.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich>=10.4.0->keybert) (0.1.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch>=1.6.0->sentence-transformers>=0.3.8->keybert) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.6.0->sentence-transformers>=0.3.8->keybert) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.6.0->sentence-transformers>=0.3.8->keybert) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.6.0->sentence-transformers>=0.3.8->keybert) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages
(from triton==2.0.0->torch>=1.6.0->sentence-transformers>=0.3.8->keybert)
(3.27.4.1)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages
(from triton==2.0.0->torch>=1.6.0->sentence-transformers>=0.3.8->keybert)
(16.0.6)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from
transformers<5.0.0,>=4.6.0->sentence-transformers>=0.3.8->keybert) (2023.6.3)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from
transformers<5.0.0,>=4.6.0->sentence-transformers>=0.3.8->keybert)
  Downloading
tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(7.8 MB)
                           7.8/7.8 MB
34.0 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers<5.0.0,>=4.6.0->sentence-
transformers>=0.3.8->keybert)
  Downloading
safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(1.3 MB)
                           1.3/1.3 MB
45.4 MB/s eta 0:00:00
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-
packages (from nltk->sentence-transformers>=0.3.8->keybert) (8.1.7)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.10/dist-packages (from torchvision->sentence-
transformers>=0.3.8->keybert) (9.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.6.0->sentence-
transformers>=0.3.8->keybert) (2.1.3)
```

```
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.4.0->sentence-transformers>=0.3.8->keybert) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->huggingface-hub>=0.4.0->sentence-
transformers>=0.3.8->keybert) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.4.0->sentence-transformers>=0.3.8->keybert) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub>=0.4.0->sentence-transformers>=0.3.8->keybert) (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch>=1.6.0->sentence-transformers>=0.3.8->keybert)
(1.3.0)
Building wheels for collected packages: keybert, sentence-transformers
  Building wheel for keybert (setup.py) … done
  Created wheel for keybert: filename=keybert-0.7.0-py3-none-any.whl size=23765
sha256=a5689850dcf059fb5433971c1b3338ea5b07e9c4f4fdb38a4b5e91a79025ee93
  Stored in directory: /root/.cache/pip/wheels/66/8d/e6/b0e2f8d883b0fd51819226f6
7ad9843e04913ce4a97241ff4b
  Building wheel for sentence-transformers (setup.py) … done
  Created wheel for sentence-transformers:
filename=sentence_transformers-2.2.2-py3-none-any.whl size=125923
sha256=787a67aba23cbc36e86ec543f8b801c585d4518e79dcd060c50a132a33a5ef18
  Stored in directory: /root/.cache/pip/wheels/62/f2/10/1e606fd5f02395388f74e746
2910fe851042f97238cbbd902f
Successfully built keybert sentence-transformers
Installing collected packages: tokenizers, sentencepiece, safetensors,
huggingface-hub, transformers, sentence-transformers, keybert
Successfully installed huggingface-hub-0.16.4 keybert-0.7.0 safetensors-0.3.3
sentence-transformers-2.2.2 sentencepiece-0.1.99 tokenizers-0.13.3
transformers-4.33.1
```

he output of this code will be a list of keywords extracted from the speech_text. These keywords are typically words or phrases that the KeyBERT algorithm considers important or representative of the content in the text. The specific keywords extracted will depend on the content of the speech_text and the behavior of the KeyBERT model.

[80]:
```
from keybert import KeyBERT
kw_extractor = KeyBERT()
keywords = kw_extractor.extract_keywords(speech_text)
for keyword in keywords:
    print(keyword[0])
```

```
Downloading (…)e9125/.gitattributes:   0%|          | 0.00/1.18k [00:00<?, ?B/s]

Downloading (…)_Pooling/config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]
Downloading (…)7e55de9125/README.md:   0%|          | 0.00/10.6k [00:00<?, ?B/s]

Downloading (…)55de9125/config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]

Downloading (…)ce_transformers.json:   0%|          | 0.00/116 [00:00<?, ?B/s]

Downloading (…)125/data_config.json:   0%|          | 0.00/39.3k [00:00<?, ?B/s]

Downloading pytorch_model.bin:   0%|          | 0.00/90.9M [00:00<?, ?B/s]

Downloading (…)nce_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]

Downloading (…)cial_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]

Downloading (…)e9125/tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

Downloading (…)okenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]

Downloading (…)9125/train_script.py:   0%|          | 0.00/13.2k [00:00<?, ?B/s]

Downloading (…)7e55de9125/vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

Downloading (…)5de9125/modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]
```

```
manipur
nehru
bjp
rajya
gandhi
```

NRCLex is used for sentiment and emotion analysis based on the NRC (National Research Council) Emotion Lexicon.

[81]:
```
!pip install nrclex
```

```
Collecting nrclex
  Downloading NRCLex-4.0-py3-none-any.whl (4.4 kB)
Requirement already satisfied: textblob in /usr/local/lib/python3.10/dist-
packages (from nrclex) (0.17.1)
INFO: pip is looking at multiple versions of nrclex to determine which version
is compatible with other requirements. This could take a while.
  Downloading NRCLex-3.0.0.tar.gz (396 kB)
                           396.4/396.4

kB 4.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.10/dist-
packages (from textblob->nrclex) (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob->nrclex) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob->nrclex) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob->nrclex)
(2023.6.3)
```

```
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob->nrclex) (4.66.1)
Building wheels for collected packages: nrclex
  Building wheel for nrclex (setup.py) … done
  Created wheel for nrclex: filename=NRCLex-3.0.0-py3-none-any.whl size=43310
sha256=9fa84cd083e892569cf8b3faf51181c8800e991a2c79b2fc3c13cdf11a09917e
  Stored in directory: /root/.cache/pip/wheels/d2/10/44/6abfb1234298806a145fd6bc
aec8cbc712e88dd1cd6cb242fa
Successfully built nrclex
Installing collected packages: nrclex
Successfully installed nrclex-3.0.0
```

emotions = text_emotion.affect_frequencies: This line extracts the affect (emotion) frequencies from the text_emotion object. The affect_frequencies attribute contains a dictionary where emotion names are keys, and their corresponding frequencies in the text are values.}

[83]:
```python
from nrclex import NRCLex

text_emotion = NRCLex(speech_text_cleaned)

emotions = text_emotion.affect_frequencies

for emotion, frequency in emotions.items():
    print(f"Emotion: {emotion}, Frequency: {frequency}")
```

```
Emotion: fear, Frequency: 0.10664993726474278
Emotion: anger, Frequency: 0.1053952321204517
Emotion: anticip, Frequency: 0.0
Emotion: trust, Frequency: 0.1329987452948557
Emotion: surprise, Frequency: 0.02258469259723965
Emotion: positive, Frequency: 0.23462986198243413
Emotion: negative, Frequency: 0.1668757841907152
Emotion: sadness, Frequency: 0.04642409033877039
Emotion: disgust, Frequency: 0.033877038895859475
Emotion: joy, Frequency: 0.06524466750313676
Emotion: anticipation, Frequency: 0.08531994981179424
```

[84]:
```python
data = {'Emotion': [], 'Frequency': []}
for emotion, frequency in emotions.items():
    data['Emotion'].append(emotion)
    data['Frequency'].append(frequency)

df_emotions = pd.DataFrame(data)

fig = px.bar(df_emotions, x='Emotion', y='Frequency', color='Emotion',
             labels={'Emotion': 'Emotion', 'Frequency': 'Frequency'},
             title='Emotion Frequencies in the Speech')
fig.show()
```

textstat library to calculate various readability scores for the cleaned text stored in the speech_text_cleaned variable.

```
[85]: !pip install textstat
```

```
Collecting textstat
  Downloading textstat-0.7.3-py3-none-any.whl (105 kB)
                           105.1/105.1

kB 1.4 MB/s eta 0:00:00
Collecting pyphen (from textstat)
  Downloading pyphen-0.14.0-py3-none-any.whl (2.0 MB)
                           2.0/2.0 MB
7.1 MB/s eta 0:00:00
Installing collected packages: pyphen, textstat
Successfully installed pyphen-0.14.0 textstat-0.7.3
```

The Flesch Reading Ease score is a measure of how easy or difficult it is to read a text. Higher scores indicate easier readability, while lower scores suggest more complex text.

The Flesch-Kincaid Grade Level is an estimate of the U.S. school grade level required to understand the text. Higher values indicate more complex text.

The SMOG Index estimates the number of years of education required to understand the text. Higher values indicate more advanced reading level requirements.

```
[88]: import textstat

flesch_score = textstat.flesch_reading_ease(speech_text_cleaned)
flesch_grade = textstat.flesch_kincaid_grade(speech_text_cleaned)
smog_index = textstat.smog_index(speech_text_cleaned)

print(f"Flesch Reading Ease Score: {flesch_score}")
print(f"Flesch-Kincaid Grade Level: {flesch_grade}")
print(f"SMOG Index: {smog_index}")
```

```
Flesch Reading Ease Score: -3018.87
Flesch-Kincaid Grade Level: 1192.8
SMOG Index: 0.0
```

NLTK (Natural Language Toolkit) to calculate and print the Pointwise Mutual Information (PMI) scores for bigrams in the cleaned text (speech_text_cleaned).

```
[87]: from nltk.collocations import BigramAssocMeasures, BigramCollocationFinder

tokens = nltk.word_tokenize(speech_text_cleaned)

bigram_measures = BigramAssocMeasures()
finder = BigramCollocationFinder.from_words(tokens)
```

```python
pmi_scores = finder.score_ngrams(bigram_measures.pmi)

for bigram, pmi in pmi_scores[:10]:
    print(f"Bigram: {bigram}, PMI: {pmi}")
```

```
Bigram: ('135', 'crore'), PMI: 11.576484346796851
Bigram: ('400', 'night'), PMI: 11.576484346796851
Bigram: ('5', 'economies'), PMI: 11.576484346796851
Bigram: ('50000', 'per'), PMI: 11.576484346796851
Bigram: ('account', 'yoga'), PMI: 11.576484346796851
Bigram: ('air', 'travel'), PMI: 11.576484346796851
Bigram: ('almost', 'eradicated'), PMI: 11.576484346796851
Bigram: ('arunachal', 'pradesh'), PMI: 11.576484346796851
Bigram: ('aspirations', 'whatever'), PMI: 11.576484346796851
Bigram: ('azad', 'hind'), PMI: 11.576484346796851
```